

CLAIMS

What is claimed is:

- 1 1. A method of managing resources in a multithreaded processor, the method
2 comprising:
3 partitioning a resource into a number of portions based upon a number of threads
4 being executed concurrently; and
5 performing resource allocation for each thread in its respective portion of the
6 resource.
- 1 2. The method of claim 1 wherein partitioning comprises:
2 sizing the corresponding portion for each thread according to a partitioning
3 scheme; and
4 marking the corresponding portion as being reserved for the respective thread.
- 1 3. The method of claim 2 wherein the size of each portion is determined based upon
2 at least one factor selected from the group consisting of a first factor indicating the
3 number of threads being executed concurrently, a second factor indicating the capacity of
4 the resource, and a third factor indicating a relative processing priority of each thread.
- 1 4. The method of claim 2 wherein marking comprises:
2 specifying the lower and upper boundaries of each portion corresponding to its
3 respective location within the resource.

1 5. The method of claim 1 further comprising:

2 initializing each portion of the resource in response to one or more signals

3 indicating a mode transition.

1 6. The method of claim 5 wherein the mode transition is invoked in response to an

2 event or condition.

1 7. The method of claim 5 wherein initializing comprises:

2 initializing a set of pointers corresponding to the respective portion.

1 8. The method of claim 7 wherein the set of pointers comprises a first pointer used to

2 keep track of entries that have been allocated in the respective portion and a second

3 pointer used to keep track of entries that have been deallocated in the respective portion.

1 9. The method of claim 1 wherein performing resource allocation for each thread

2 comprises:

3 performing stall computation for each thread to determine whether the respective

4 portion has sufficient available entries to allocate a number of entries required for the

5 execution of one or more instructions from the respective thread; and

6 allocating the number of entries required in the respective portion if the respective

7 portion has sufficient available entries.

1 10. The method of claim 9 wherein performing stall computation for each thread is

2 done in parallel with performing stall computation for another thread.

1 11. The method of claim 9 wherein performing stall computation for each thread and
2 performing stall computation for another thread are multiplexed.

1 12. The method of claim 9 wherein allocating the number of entries required for each
2 thread is done in parallel with allocating a number of entries required for another thread.

1 13. The method of claim 9 wherein allocating the number of entries required for each
2 thread and allocating a number of entries required for another thread are multiplexed.
3

1 14. The method of claim 9 wherein performing stall computation for each thread
2 comprises:
3 determining the number of entries to be allocated for the one or more instructions
4 from the respective thread;
5 determining a number of entries available in the respective portion; and
6 comparing the number of entries to be allocated with the number of entries
7 available in the respective portion.

1 15. The method of claim 14 further comprising:
2 activating one or more stall signals if the number of entries required exceeds the
3 number of entries available in the respective portion, the one or more stall signals
4 indicating that the one or more instructions from the respective thread cannot be executed
5 due to insufficient available resource in the respective portion.

1 16. The method of claim 14 wherein determining the number of entries to be allocated
2 for the one or more instructions comprises:
3 determining the type of the one or more instructions; and
4 determining whether the resource is needed to execute the one or more
5 instructions based upon the type of the one or more instructions.

1 17. The method of claim 16 wherein the number of entries to be allocated is greater
2 than the number of entries needed to execute the one or more instructions.

1 18. The method of claim 14 wherein determining the number of entries available
2 comprises:
3 comparing the value of the first pointer with the value of the second pointer to
4 determine the number of entries that are available for allocation.

1 19. The method of claim 18 further comprising:
2 wrapping the first pointer when it is advanced past the end of the respective
3 portion.
4

1 20. The method of claim 19 including:
2 updating a wrap bit to indicate that the first pointer is wrapped around.
3

1 21. The method of claim 18 further comprising:
2 wrapping the second pointer when it is advanced past the end of respective
3 portion.

1 22. The method of claim 21 including:

2 updating a wrap bit to indicate that the second pointer is wrapped around.

3

1 23. A method of managing a resource in a multithreaded processor, the method

2 comprising:

3 detecting a signal indicating a processing mode;

4 performing resource allocation according to a multithread scheme if the processing

5 mode is multithreading; and

6 performing resource allocation according to a single thread scheme if the

7 processing mode is single threading

1 24. The method of claim 23 wherein the signal indicating the processing mode is

2 updated in response to an occurrence of an event or a condition.

3

1 25. The method of claim 23 wherein performing resource allocation according to the

2 single thread scheme comprises:

3 determining which thread is active; and

4 assigning all of the resource to the thread which is active.

1 26. The method of claim 25 including:

2 performing resource allocation for the thread which is active.

1 27. The method of claim 26 wherein performing resource allocation for the active

2 thread comprises:

3 initializing pointers of the active thread in response to one or more signals
4 indicating a mode transition;
5 receiving a set of instructions from the active thread;
6 determining whether the resource has sufficient available entries to allocate for the
7 set of instructions from the active thread; and
8 allocating an amount of entries needed for the set of instructions if the resource has
9 sufficient available entries to allocate.

1 28. The method of claim 27 further comprising:
2 activating one or more stall signals for the active thread if the resource does not
3 have sufficient available entries to allocate, the one or more stall signals indicating that the
4 set of instructions cannot be executed due to insufficient resource.

1 29. The method of claim 27 wherein determining whether the resource has sufficient
2 available entries comprises:
3 computing the amount of entries to be allocated for the set of instructions;
4 computing the amount of available entries in the resource; and
5 comparing the amount to be allocated with the amount of available entries.

1 30. The method of claim 23 wherein performing resource allocation according to the
2 multithread scheme comprises:
3 partitioning the resource into a number of portions corresponding to a number of
4 threads being executed concurrently; and

5 performing resource allocation for each thread in its respective portion of the
6 resource.

1 31. The method of claim 30 wherein the size of each portion is predetermined.

1 32. The method of claim 30 wherein the size of each portion is determined based upon
2 at least one factor selected from the group consisting of a first factor indicating a number
3 of active threads executed concurrently, a second factor indicating the capacity of the
4 resource, and a third factor indicating a relative processing priority assigned to each
5 thread.

1 33. The method of claim 30 wherein performing resource allocation comprises:
2 initializing pointers corresponding to each portion in response to one or more
3 signals indicating a mode transition;
4 receiving a set of instructions from the respective thread; and
5 determining whether the respective portion has sufficient available entries to
6 allocate for the set of instructions from the respective thread.

1 34. The method of claim 33 including:
2 activating one or more stall signals for the respective thread if the respective
3 portion does not have sufficient available entries to allocate.

1 35. The method of claim 34 further comprising:
2 determining whether the set of instructions belongs to the respective thread; and

3 allocating an amount of entries needed for the set of instructions in the respective
4 portion if the set of instructions belongs to the respective thread and the one or more stall
5 signal for the respective thread is not activated.

1 36. The method of claim 35 wherein determining whether the set of instructions
2 belongs to the respective thread comprises:
3 examining the value of a thread bit associated with the respective instruction, the
4 value of the thread bit indicating which thread the respective instruction belongs.

1 37. The method of claim 33 wherein determining whether the respective portion has
2 sufficient available entries comprises:
3 computing an amount of entries to be allocated for the set of instructions;
4 computing an amount of available entries in the respective portion; and
5 comparing the amount of entries to be allocated with the amount of available
6 entries.

1 38. The method of claim 34 including:
2 stalling further fetching of instructions from the respective thread if the one or
3 more stall signals for the respective thread is activated.

1 39. An apparatus for managing a resource in a multithreaded processor, the apparatus
2 comprising:
3 partition logic to partition the resource into a number of portions corresponding to
4 a number of threads being executed concurrently; and

5 resource control logic to perform resource allocation for each thread in its
6 respective portion of the resource.

1 40. The apparatus of claim 39 wherein the size of each portion is predetermined.

1 41. The apparatus of claim 39 wherein the size of each portion is determined based
2 upon at least one factor selected from the group consisting of a first factor indicating the
3 number of threads being executed concurrently, a second factor indicating the capacity of
4 the resource, and a third factor indicating a relative processing priority of each thread.

1 42. The apparatus of claim 39 further comprising:
2 initialization logic to initialize pointers corresponding to each respective portion in
3 response to one or more signals indicating a mode transition.

1 43. The apparatus of claim 42 wherein the initialization logic updates the pointers to
2 point to the respective portion of the resource.

1 44. The apparatus of claim 43 wherein the pointers comprise a first pointer used to
2 keep track of entries that have been allocated in the respective portion and a second
3 pointer used to keep track of entries that have been deallocated in the respective portion.

1 45. The apparatus of claim 39 wherein the resource control logic comprises:

2 stall computation logic to determine whether the respective portion has sufficient
3 available entries to allocate an amount of entries for a set of instructions from the first
4 thread; and

5 allocation logic to allocate the amount of entries for the set of instructions in the
6 respective portion if the respective portion has sufficient available entries to allocate.

1 46. The apparatus of claim 39 wherein stall computations for the respective threads are
2 performed in parallel and resource allocation for the respective threads are performed in
3 parallel.

1 47. The apparatus of claim 39 wherein stall computations for the respective threads are
2 performed in parallel and resource allocation for the respective threads are performed in a
3 multiplexed manner.

1 48. The apparatus of claim 45 wherein the stall computation logic comprises:
2 resource requirement logic to compute an amount of entries to be allocated for the
3 set of instructions from the respective thread;
4 resource availability logic to determine an amount of entries available in the
5 respective portion; and
6 comparison logic to compare the amount of entries to be allocated with the
7 amount of entries available.

1 49. The apparatus of claim 48 further comprising:

2 stall activation logic to activate one or more stall signals if the amount to be
3 allocated exceeds the amount of entries available, the one or more stall signals indicating
4 that the set of instruction from the respective thread cannot be executed due to insufficient
5 available resource in the respective portion.

1 50. The apparatus of claim 48 wherein the resource availability logic comprises:
2 first tracking logic to keep track of the amount of entries in the respective portion
3 that have been allocated; and
4 second tracking logic to keep track of the amount of entries in the respective
5 portion that have been deallocated.

1 51. An apparatus for controlling usage of a resource in a multithreaded processor, the
2 apparatus comprising:
3 detection logic to detect a signal indicating a processing mode; and
4 a control circuit to perform resource allocation according to a single thread
5 scheme if the processing mode is single threading and to perform resource allocation
6 according to a multithread scheme if the processing mode is multithreading.

1 52. The apparatus of claim 51 wherein the control circuit comprises resource partition
2 logic to partition the resource into a number of portions according to a number of threads
3 being executed concurrently.

1 53. The apparatus of claim 52 wherein the resource partition logic assigns all of the
2 resource to the thread that is active if the processing mode is single threading.

1 54. The apparatus of claims 52 wherein the resource partition logic assigns a portion
2 of the resource to each of the threads being executed concurrently if the processing mode
3 is multithreading.

1 55. The apparatus of claim 52 wherein the control circuit further comprises:
2 resource allocation logic to perform resource allocation based upon the number of
3 threads being executed concurrently.

1 56. The apparatus of claim 55 wherein the control circuit comprises:
2 initialization logic to initialize the resource based upon the number of threads being
3 executed concurrently.

1 57. The apparatus of claim 55 wherein the resource allocation logic comprises:
2 resource requirement logic to compute an amount of entries to be allocated for a
3 set of instructions;
4 resource availability logic to compute an amount of available entries in the
5 resource; and
6 allocate logic to allocate the amount of entries for the set of instructions in the
7 resource if the amount of available entries is sufficient.

1 58. The apparatus of claim 57 wherein the resource availability logic computes the
2 amount of available entries with respect to the entire resource if the processing mode is
3 single threading.

1 59. The apparatus of claim 57 wherein the resource availability logic computes the
2 amount of available entries with respect to the portion assigned to the respective thread if
3 the processing mode is multithreading.

1 60. The apparatus of claim 57 wherein the allocate logic comprises logic to allocate
2 the amount of entries in the corresponding portion if the set of instructions belongs to the
3 respective thread and the corresponding portion has enough available entries.

1 61. The apparatus of claim 57 including:
2 stall activation logic to activate at least one stall signal if the amount of entries
3 available is not sufficient.

1 62. A processor comprising:
2 an instruction delivery engine to store and fetch instructions either from one or
3 more threads based upon a current processing mode; and
4 an allocator to receive instructions from the instruction delivery engine and to
5 perform allocation in a resource based upon the current processing mode.

1 63. The processor of claim 62 wherein the allocator assigns the entire resource to the
2 thread that is active if the current processing mode is single threading.

1 64. The processor of claim 62 wherein the allocator assigns a portion of the resource
2 to each of the threads running concurrently if the current processing mode is
3 multithreading.

1 65. The processor of claim 63 wherein the allocator allocates an amount of entries for
2 the instructions from the active thread in the resource if the resource has sufficient
3 available entries and wherein the allocator activates at least one stall signal if the resource
4 does not have sufficient available entries.

1 66. The processor of claim 64 wherein the allocator allocates an amount of entries for
2 the instructions from each respective thread in the respective portion if the respective
3 portion has sufficient available entries and wherein the allocator activates at least one stall
4 signal if the respective portion does not have sufficient available entries.

1 67. The processor of claim 66 wherein the instruction delivery engine uses the at least
2 one stall signal to perform its corresponding function.

1 68. The processor of claim 67 wherein the instruction delivery engine re-fetches the
2 stalled instructions in the respective thread to the allocator if the at least one stall signal is
3 activated.

1 69. The processor of claim 67 wherein the instruction delivery engine fetches a
2 subsequent instruction from another thread to the allocator if the at least one stall signal
3 for the respective thread is activated and said another thread is not stalled.

1 70. The processor of claim 67 wherein the instruction delivery engine fetches an
2 invalid instruction to the allocator if the stall signal for the respective thread is activated.

1 71. An apparatus for managing a resource in a multithreaded processor, the apparatus
2 comprising:

3 means for assigning a portion of the resource to each of a plurality of threads being
4 executed concurrently in the multithreaded processor; and

5 means for performing resource allocation for each respective thread in its
6 respective portion of the resource.

1 72. The apparatus of claim 71 further comprising:

2 means for initializing each respective portion in response to a signal indicating a
3 mode transition.

1 73. The apparatus of claim 72 wherein means for initializing comprises:

2 means for setting a corresponding set of pointers to point to the respective portion.

1 74. The apparatus of claim 71 wherein means for performing resource allocation for
2 the respective thread comprises:

3 means for performing stall computation for the respective thread to determine
4 whether the respective portion has sufficient available entries to allocate an amount of
5 entries for the execution of a set of instructions from the respective thread; and

6 means for allocating the amount of entries in the respective portion for the set of
7 instructions if the respective portion has sufficient available entries to allocate.

1 75. The apparatus of claim 74 wherein means for performing stall computation
2 comprises:

3 means for determining the amount of entries to be allocated for the set of
4 instructions;
5 means for determining an amount of entries available in the respective portion; and
6 means for comparing the amount of entries to be allocated with the amount of
7 entries available.

1 76. The apparatus of claim 75 further comprising:

2 means for activating at least one stall signal if the amount to be allocated exceeds
3 the amount of entries available, the at least one stall signal indicating that the set of
4 instructions from the respective thread cannot be executed due to insufficient available
5 resource.

1 77. The apparatus of claim 75 wherein means for determining the amount of entries
2 available comprises:

3 means for keeping track of the amount of entries in the respective portion that
4 have been allocated; and

5 means for keeping track of the amount of entries in the respective portion that
6 have been deallocated.

1 78. An apparatus for controlling usage of a resource, the apparatus comprising:

2 detection means for detecting a signal indicating a processing mode; and

3 control means for performing resource allocation according to a single thread
4 scheme if the processing mode is single threading and for performing resource allocation
5 according to a multithread scheme if the processing mode is multithreading.

1 79. The apparatus of claim 78 wherein the control means comprises:
2 partition means for partitioning the resource into a number of portions based upon
3 a number of threads being executed concurrently.

1 80. The apparatus of claim 79 wherein the control means further comprises:
2 allocation means for allocating the resource based upon the number of threads
3 being executed concurrently.

1 81. The apparatus of claim 80 wherein the control means comprises:
2 initialization means for initializing the resource based upon the number of threads
3 being executed concurrently.

1 82. The apparatus of claim 80 wherein the allocation means comprises:
2 requirement computing means for computing an amount of entries to be allocated
3 for a set of instructions;
4 availability computing means for computing an amount of available entries in the
5 resource; and
6 means for allocating the amount of entries in the resource if the amount of
7 available entries is sufficient.

add
pg 7

